Assignment 02

Dubin Car

Grid

At the very begin, the continuous space should be split into grids. The grid size should ensure that: the dubin car will keep in the grid when rotating at the center of the grid.

Also, an error radius should be considered - the dubin car is supposed to be arrived at the destination when distance bewteen them is less or equal to the error radius. Set the error radius as 5 (variable `errorRadius` in file `ZhichengDubinCarController.java`).

Based on aformentioned discussions and the size of dubing car, the minimum size of the grid is:

$$\begin{aligned}&\left(\sqrt{(\frac{width}{2}+thickness)^2+(length-wheelRadius)^2}+errorRadius\right)\times 2\\ =&\left(\sqrt{(\frac{20}{2}+4)^2+(40-5)^2}+5\right)\times 2\approx 85.392\end{aligned}$$

Set the grid size as 100 for easy calculation (variable `gridSize` in file `ZhichengDubinCarController.java`).

Each grid has an index x and and an index y, which are nonnegative integers. They constitute the state (file `ZhichengState.java`) of the problem (file `ZhichengProblem.java`).

Neiborhoods of a grid/state is restricted as appressed grids on the left, right, up and down which does NOT contain any part of obsticle. To illustrate such restriction, file `ZhichengObstacle.java` converts (expands) obstacles to the grid system.

Problem & Planner

Based on interface `PlanningProblem` and class `CBPlannerAStar` (not included here), "problem" and "planner" are created for this assignment. They are:

- file `ZhichengProblem.java`
- file `ZhichengCBPlannerAStar.java`

In file `ZhichengProblem.java`, function `getNeighbors` shows how to calculate variable `costFromStart` and varialble `estimatedCostToGoal`.

Controller

The controller of the dubin car is defined in file `ZhichengDubinCarController.java`. It provides the following procedure:

1. The optimal solution (shortest path) in grid level is generated by function `init` (A* algorithm).
2. The instruction - a set of steps (file `ZhichengStep`) - is converted by the optimal solution. Function `solutionToInstruction` shows the detail:
    - move from the origin to the center of the first grid
    - move between grids.
    - move from the center of the last grid to the destination
3. Function `move` uses the instruction to control the car.

Unfortunately, there are still some bugs in #2 which cause infinite loop sometimes. (I have tried my best to fix them but failed.) By this, the controller has not been avaliable for program `carGUI`.

But there are still some achievements: execute function `main`, and a set of coordinates will show a path of the optimal solution of `Scene 3`.

Unicycle

Grid

Like the dubin car, set the error radius as 5. The minimum size of the grid is:

$$\left(\sqrt{(\frac{width}{2})^2+(\frac{length}{2})^2}+errorRadius\right)\times 2$$
$$=\left(\sqrt{(\frac{30}{2})^2+(\frac{16}{2})^2}+5\right)\times 2\approx 44$$

Set the grid size as 50 for easy calculation.

Problem & Planner

(same as the dubin car)

Controller

The procedure is similar as the dubin car. The only difference is the way of control.